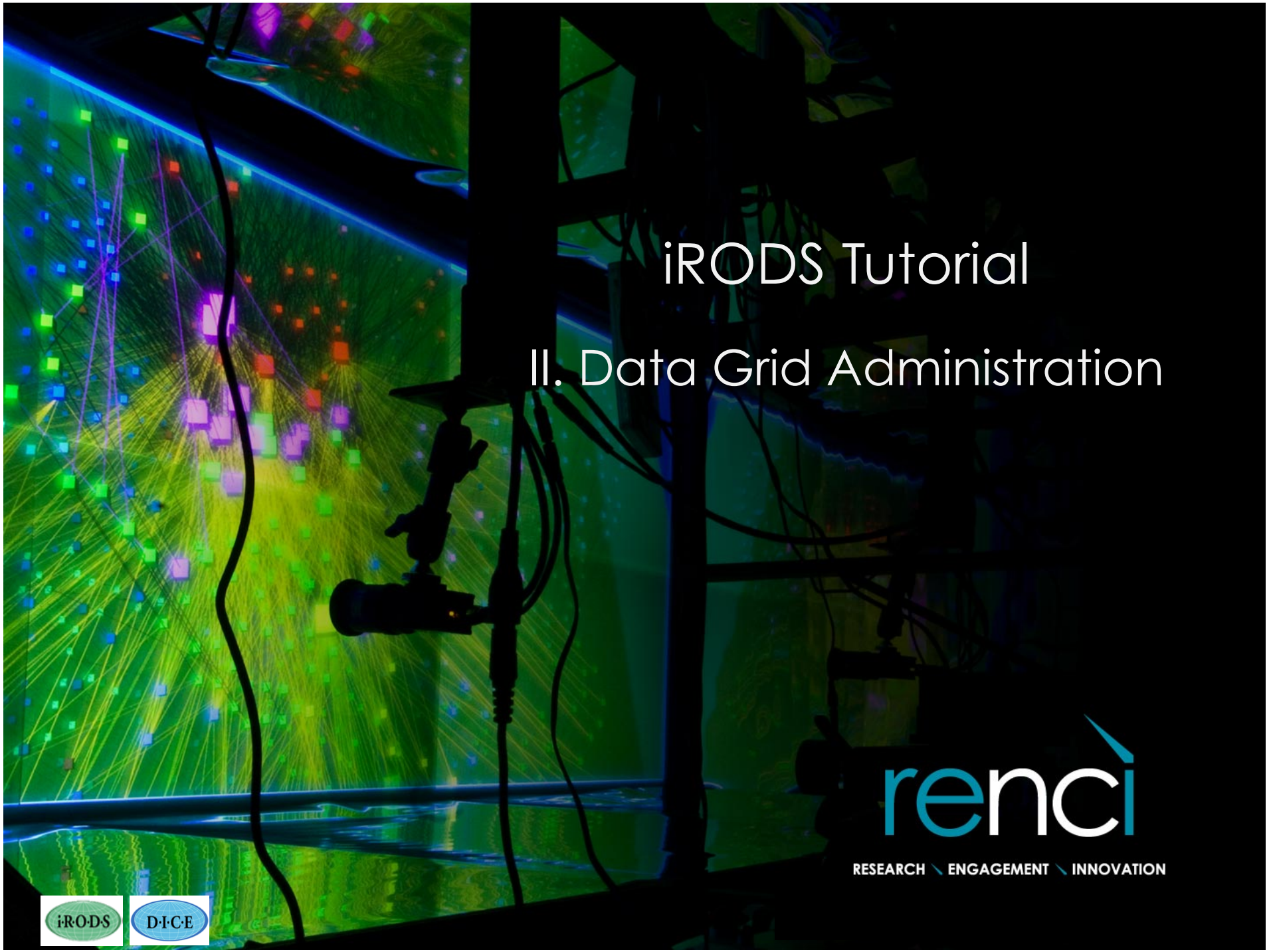

iRODS Tutorial
II. Data Grid Administration
renci
RESEARCH ENGAGEMENT INNOVATION
i·R·O·D·S
D·I·C·E

iRODS Tutorial Preview

I. iRODS Getting Started
   - unix client
   - usage

II. iRODS Data Grid Administration
   - installing server and iCAT
   - setting up users
   - adding new resources to a data grid/zone
   - federating with other grids/zones, remote users
   - microservices and rules for policy implementation and enforcement

II. iRODS Data Grid Administration

iRODS Info

- Main page: http://www.irods.org
- Chat list: irods-chat@irods.org
- iRODS Documentation:
  https://www.irods.org/index.php/Documentation
- On-line tutorial:
  https://www.irods.org/index.php/Tutorial

iRODS Books

Available from Amazon

- iRODS Primer: integrated Rule-Oriented Data System (Synthesis Lectures on Information Concepts, Retrieval, and Services)
  http://www.amazon.com/dp/1608453332

- The integrated Rule-Oriented Data System (iRODS) Micro-service Workbook
  http://www.amazon.com/dp/1466469129

iRODS Download

- Downloads link on the iRODS main page: https://www.irods.org/download.html
  - BSD license
  - registration/agreement

- SVN repository contains patches, pre-release features:

  svn checkout svn://irodssvn.ucsd.edu/trunk

- Untar irods3.1.tgz
  - cd into a directory where you want to install iRODS, eg $HOME/tutorial
  - Untar: tar –zxvf irods3.1.tgz
  - cd into iRODS

iRODS Installation

- cd ~/tutorial/iRODS (iRODS directory)
- Run the install script: ./irodssetup
- Can install three main components using irodssetup:
  1. an iRODS **server** (iCAT-enabled or not)
  2. the iCAT catalog metadata **database**
  3. 'icommands' – the **unix client**
- Install an iCAT-enabled iRODS server here…

iRODS iCAT-enabled Server (IES) Installation

- ./irodssetup [no response takes default value]

  Include additional prompts for advanced settings [no]? yes

  Build an iRODS server [no]? yes

  Make this Server ICAT-Enabled [yes]? yes

  iRODS zone name [tempZone]? myZone

  iRODS login name [rods]? rods_admin

  Password [rods]? ********

  Port [1247]? 1257

  Starting Server Port [20000]?

  Ending Server Port [20199]?

  iRODS database name [ICAT]?

Name your own zone, admin user, password, port number.

iRODS iCAT-enabled Server (IES) Installation

- ./irodssetup [no response takes default value]

  iRODS DB password scramble key [123]?

  Resource name [demoResc]? myResc

  Directory [/home/user/leesa/iRODS/Vault]?
  /home/user/leesa/Vault

  Download and build a new Postgres DBMS [yes]?

  New Postgres directory? /home/user/leesa

  New database login name [leesa]?

  Password? *****

  PostgreSQL version [postgresql-9.0.3.tar.gz]?

  ODBC version [unixODBC-2.2.12.tar.gz]?

  Port [5432]? 5433

Name your own resource, vault path name (an existing directory), DB admin, DB port.

iRODS iCAT-enabled Server (IES) Installation
(continued)

- ./irodssetup [no response takes default value]

  Include GSI [no]?

  Include the NCCS Auditing extensions [no]?

  Save configuration (irods.config) [yes]?

  Start iRODS build [yes]?

- This also builds the icommands client.

iRODS Post-Install

- Configuration parameters saved in iRODS/config/irods.config
- Install logs in iRODS/installLogs/
- Server log in iRODS/server/log/
- Put the icommands in your PATH

```
> cd $HOME/bin
> ln –s /home/user/leesa/iRODS/clients/icommands/bin icommands
```

- Environment file $HOME/.irods/.irodsEnv is created automatically

.irodsEnv file – the data grid environment

Example for a RENCI demo data grid
(installed on host ischia.renci.org)

```
# iRODS server host name:
irodsHost 'ischia.renci.org'

# iRODS server port number:
irodsPort 1257

# Default storage resource name:
irodsDefResource 'myResc'

# Home directory in iRODS:
irodsHome '/myZone/home/rods_admin'

# Current directory in iRODS:
irodsCwd '/myZone/home/rods_admin'

# Account name:
irodsUserName 'rods_admin'

# Zone:
irodsZone 'myZone'
```

.irodsEnv

- Contains the environment of the grid you want to contact OR the grid you are running
- Use multiple environment files to choose from among many grids (only one at a time has the name .irodsEnv)
- Do NOT use multiple .irodsEnv files in the unix account running a grid
- Can run multiple data grids on a host, but to avoid contention…
  - Keep separate unix accounts to run the separate data grids
  - Never change the .irodsEnv file of a unix account running a grid
  - Use different port number sets (for iRODS server and the iCAT DB) for each data grid

Setting Up New Users

- Use iadmin
- Two steps: mkuser and moduser (for a password)

```
iadmin> mkuser  user1  rodsuser
iadmin> moduser  user1  password  *****
```

- Use iadmin to see what user types are possible

```
iadmin> lt
        zone_type
        user_type
        data_type
        resc_type
        action_type
        rulexec_type
        access_type
        object_type
        resc_class
        coll_map
        auth_scheme_type
```

Token List

```
iadmin> lt user_type
        rodsgroup
        rodsadmin
        rodsuser
        domainadmin
        groupadmin
        storageadmin
        rodscurators
```

Possible values of token "user_type"

iRODS non-iCAT Server Installation

- An admin user must set up the secondary resource
- iCAT server must know of the secondary resource. On the host running the data grid, run mkresc (part of iadmin):

  mkresc Name Type Class Host [Path]

  >iadmin mkresc myResc2
  "unix file system" cache
  host2.renci.org
  /projects/irods/myVault

- Bring up the new server on the second host:

  >./irodssetup [no response takes default value]

  Include additional prompts for advanced settings [no]? yes

  Build an iRODS server [no]? yes

  Make this Server ICAT-Enabled [yes]? no

  Host running iCAT-enabled iRODS server? ischia.renci.org
  Resource name? myResc2

iRODS non-iCAT Server Installation

- ./irodssetup continued… [no response takes default value]

  Resource storage area directory [/home/user/leesa/iRODS/Vault]? /projects/irods/myVault

  Existing iRODS admin login name [rods]? rods_admin

  Password [******]?

  iRODS zone name [tempZone]? myZone

  Port [1257]?

  Starting Server Port [20000]?

  Ending Server Port [20199]?

  Include GSI [no]?

  Include the NCCS Auditing extensions [no]?

  Save configuration (irods.config) [yes]?

  Start iRODS build [yes]?

Usually the admin account for this server will be the same account as for the iCAT-Enabled Server (IES).

iRODS control

- ./irodsctl
  - start
  - stop
  - restart

  Start/stop/restart the iRODS server and the iCAT

  - istart
  - istop
  - irestart

  Start/stop/restart the iRODS server but not the iCAT

iadmin – administrative functions

- h for help
- quit to exit
- Add new users, modify passwords, add new resources, federate to remote zones, create resource groups,…
  - mkresc/rmresc
  - mkuser/rmuser, moduser (modify passwords)
  - mkzone/rmzone, modzone (for federation)
- Information on users, resources, tokens, etc
  - lt (el-tee)
  - lu, lr, lz,…

Federation between data grids

- https://www.irods.org/index.php/Federation_Administration
- Zone A acknowledges Zone B: iadmin mkzone B remote Host:Port
- Zone B acknowledges Zone A: iadmin mkzone A remote Host:Port
- Zone A adds remote users: iadmin mkuser some_user#B
- Zone B adds remote users: iadmin mkuser other_user#A
- User can see resources in remote zone A: ilsresc –z A

Admin users from one grid won't necessarily be admin users on the other grid.

Removing/deleting data or resources

Administrator activities

- "irm /zone/home/user/file1" moves file1 to /zone/trash/user/file1
  Not physically removed from disk
- "irm –f /zone/home/user/file1" physically deletes file1
- When removing a resource, it must be empty
  - If files are in the trash directory, resource is not empty
- To delete old users' files for removing a resource
  - Admin user can use ichmod –M in admin mode
  - Admin user can set environment variable clientUserName as the user whose files are obsolete and need to be removed from the iCAT

Administrative Rights

- -M option for some commands: ichmod
- Admin user can acquire other iRODS user's identity
  - iinit as admin user (say "rods")
  - set environment variable clientUserName as other user:

    setenv clientUserName baretto
  - "ienv" shows same irodsUserName (rods), however rights and permissions on the grid are now as the other user
  - to get back to "rods" identity: unsetenv clientUserName
- Some rules and queries are restricted to admin users
- Strict ACL exceptions for admins

ireg – the administrator's side

Get data into iRODS without making an additional copy or moving it

Example: Directory /vault2/state-data contains state LiDAR data that we now want in an iRODS repository… *without* moving or copying it

1. /vault2/state-data is mounted on the iRODS server host

2. Data admin sets up existing directory as an iRODS resource:
   > iadmin mkresc stateResc "unix file system" cache
   ischia.renci.org /vault2/state-data

3. User registers existing data into iRODS iCAT
   > ireg -C /vault2/state-data /myZone/home/rods_admin/state-data
   (-f option for picking up unregistered files)

*Register incoming files rigorously OR modify a directory **only** through iRODS once it has been registered to keep the iCAT consistent with the directory.*

iquest

- Query iCAT of remote zone A: iquest –z A …

- SQL logging is possible to see actual SQL queries generated using iquest
  - Edit scripts/perl/irodsctl.pl - uncoment the line
    $spLogSql = "1";
  - ./irodsctl irestart
  - Logged into iRODS/server/log files

MSO: Microservice Objects

Supporting realizable objects

- Drivers support connections to external data
- Done through microservices
  - msoDrivers module
  - two microservice drivers for each protocol (get & put)
- Instantiated through a compound resource
- Symbolic links implemented for http and Z39.50
- Admin users can implement new drivers: See **How to Create a New MSO Type** at

  https://www.irods.org/index.php/How_to_Create_a_New_MSO_Type

Symbolic Links to an http Source – the administrator's side

- Requires libcurl
- Turn on the msoDrivers module
  - yes in info.txt
  - edit Makefile in iRODS/modules/msoDrivers - uncomment the line:
    MSOHTTP = 1
- Stop server, recompile, and restart
  ./irodsctl istop
  ./irodssetup
  (irodssetup restarts the server)

Symbolic Links to an http Source

- Admin user sets up the mso resource and group

> iadmin

> mkresc httpResc mso compound ischia.renci.org ← creating an mso resource

> atrg httpGroup httpResc ← creating a resource group

> atrg httpGroup myResc ← add an existing resource of class "cache"

- User registers external data

> ireg -D mso -R httpResc -G httpGroup

"//http://www.renci.org/~leesa/slides/irods-intro.pdf"

/myZone/home/rods_admin/slides/irods-intro.pdf

Attention: additional "//" here!

Sys admin may need to tweak iRODS/server/config/irodsHost file.

Symbolic Links to an http Source

- User registers external data
- Data is then available to anyone with authorization to access the user's collection
- iget causes a replica to be made in the cache resource of the mso group (httpGroup in the preceding example)

S3 Resources – Cloud Management

See https://www.irods.org/index.php/S3_Resource

1. Set up an Amazon S3 resource
   - http://aws.amazon.com/s3/
   - You will need both the Access Key ID and the Secret Access Key

2. Download and build the libs3 library:

   http://libs3.ischo.com.s3.amazonaws.com/index.html

S3 Resources – Cloud Management

3. Edit iRODS/config/config.mk
   - Uncomment the line: AMAZON_S3=1
   - Define the s3 libraries header directories, for example:

     S3_LIB_DIR=/home/leesa/amazon/libs3-2.0/build/lib

     S3_HDR_DIR=/home/leesa/amazon/libs3-2.0/build/include

4. Add path to the S3 library to the LD_LIBRARY_PATH environment variable:

   setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:
   /home/leesa/amazon/libs3-2.0/build/lib

5. Rebuild the server

   ./irodsctl istop

   ./irodssetup (or gmake and then ./irodsctl istart)

Configuring an Amazon S3 Resource

6. Set up authentication to your Amazon resource
   - In server/config, use the file s3Auth.template as the template for the s3Auth file
   - cp s3Auth.template s3Auth
   - Edit s3Auth as indicated in template file: add S3_ACCESS_KEY_ID and S3_SECRET_ACCESS_KEY that you got from Amazon

7. Create an S3 compound resource

   > iadmin

   > mkresc s3Resc s3 compound ischia.renci.org /rodsVault

   > atrg s3Group s3Resc ← create resource group

   > atrg s3Group comp523Resc ← add resource of class "cache"

Cloud Resource

- Admin creates the S3 resource - see the S3 resource and group:

```
> ilsresc
        msoResc2
        demoResc
        cacheResc
        bundleResc
        comp523Resc
        s3Resc
        stateResc
        compResc
        cpsresc
        s3Group (resource group)
        msoRescGroup (resource group)
```

- Any user can ingest and access data there (unless your own policy forbids it)

```
> iput -f -K -R s3Resc irods-intro.pptx
```

- Cloud data is now managed by iRODS

Rules

- New rule engine with 3.0
- See https://www.irods.org/index.php/Changes_and_Improvements_to_the_Rule_Language_and_the_Rule_Engine
- Implement data policy
  - Retention, distribution, arrangement
  - Authenticity, provenance, description
  - Integrity, replication, synchronization
  - Deletion, trash cans, versioning
  - Archiving, staging, caching
  - Authentication, authorization, redaction
  - Access, approval, IRB, audit trails, report generation
  - Assessment criteria, validation
  - Derived data product generation, format parsing

Policy is the clear statement of how data will be managed over its life cycle.

Microservices

- C code
- the unit of work within iRODS
- called by rules
- composed into workflows by rules

Running Rules

- triggered by events/policy points
- contained in the (distributed) rule base:
  - iRODS/server/config/reConfigs/core.re
  - first rule with satisfied condition is executed; others are skipped
- can be run with irule - *manual execution*
- delayed execution
  - iqstat
  - iqmod

irule – to run a rule manually

- Example rules to tweak and run in the software distribution at iRODS/clients/icommands/test/rules3.0

- Some rules can only be run by admin users

Policy Enforcement Points

- Locations within iRODS framework where an event or state (of the environment) prompts a rule to execute
  - Each action may involve multiple policy enforcements points
- Policy enforcement points
  - Pre-action policy (eg, selection of storage location)
  - Execution/action policy (eg, file deletion)
  - Post-action policy (eg, create secondary data products)
- Actions (trigger rules) are contained in
  iRODS/server/config/reConfigs/core.re

Policy Enforcement Points (71)

ACTION	PRE-ACTION POLICY	POST-ACTION POLICY
acCreateUser	acPreProcForCreateUser	acPostProcForCreateUser
acDeleteUser	acPreProcForDeleteUser	acPostProcForDeleteUser
acGetUserbyDN	acPreProcForModifyUser	acPostProcForModifyUser
acTrashPolicy	acPreProcForModifyUserGroup	acPostProcForModifyUserGroup
acAclPolicy	acChkHostAccessControl	acPostProcForDelete
acSetCreateConditions	acPreProcForCollCreate	acPostProcForCollCreate
acDataDeletePolicy	acPreProcForRmColl	acPostProcForRmColl
acRenameLocalZone	acPreProcForModifyAVUMetadata	acPostProcForModifyAVUMetadata
acSetRescSchemeForCreate	acPreProcForModifyCollMeta	acPostProcForModifyCollMeta
acRescQuotaPolicy	acPreProcForModifyDataObjMeta	acPostProcForModifyDataObjMeta
acSetMultiReplPerResc	acPreProcForModifyAccessControl	acPostProcForModifyAccessControl
acSetNumThreads	acPreprocForDataObjOpen	acPostProcForOpen
acVacuum	acPreProcForObjRename	acPostProcForObjRename
acSetResourceList	acPreProcForCreateResource	acPostProcForCreateResource
acSetCopyNumber	acPreProcForDeleteResource	acPostProcForDeleteResource
acVerifyChecksum	acPreProcForModifyResource	acPostProcForModifyResource
acCreateUserZoneCollections	acPreProcForModifyResourceGroup	acPostProcForModifyResourceGroup
acDeleteUserZoneCollections	acPreProcForCreateToken	acPostProcForCreateToken
acPurgeFiles	acPreProcForDeleteToken	acPostProcForDeleteToken
acRegisterData	acNoChkFilePathPerm	acPostProcForFilePathReg
acGetIcatResults	acPreProcForGenQuery	acPostProcForGenQuery
acSetPublicUserPolicy	acSetReServerNumProc	acPostProcForPut
acCreateDefaultCollections	acSetVaultPathPolicy	acPostProcForCopy
acDeleteDefaultCollections		acPostProcForCreate

Strict ACL Policy – acAclPolicy rule

- In iRODS/server/config/reConfigs/**core.re**, replace the default AclPolicy rule with one that call the microservice to set the access control to strict:

```
#acAclPolicy { }
acAclPolicy {msiAclPolicy("STRICT"); }
```

- This disallows perusal of the collections by users without read permission.

- See the difference between compZone and norZone

Format of a Rule

```
Rule_name{
    microservice1(…,*A,…,*B);
    microservice2(*A,…);
}
INPUT *A="first_input", *B="second_input"
OUTPUT ruleExecOut
```

(*A and *B are here just for illustrative purposes…)

"ruleExecOut" is a structure managed by iRODS.

OR

```
Rule_name(*arg)  {
    on(exp)  {
      microservice1(…,*arg);
      microservice2(…);
    }
}
INPUT null
OUTPUT ruleExecOut
```

- A rule can take arguments.
- A rule can be executed conditionally.
- Use "null" if there are no input parameters.

Example Rules

- listMS.r (lists all available microservices)

```
ListAvailableMS {
    msiListEnabledMS(*KVPairs);
    writeKeyValPairs("stdout", *KVPairs, ": ");
}
INPUT null
OUTPUT ruleExecOut
```

- Tutorial: tweak and run some of the rules in
  http://www.renci.org/~leesa/rules/

- More examples rules in
  iRODS/clients/icommands/test/rules3.0

rulemsiExecCmd.r

Run an executable script

```
myTestRule {
#Input parameters are:
#  Command to be executed located in directory irods/server/bin/cmd
#  Optional command argument
#  Optional host address for command execution
#  Optional hint for remote data object path, command is executed on host
#  where the file is stored
#  Optional flag.  If > 0, use the resolved physical data object path as first argument
#Output parameter is:
#  Structure holding status, stdout, and stderr from command execution
#Output:
#  Command result is
#  Hello world written from irods
#
   msiExecCmd(*Cmd,*Arg,"null","null","null",*Result);
   msiGetStdoutInExecCmdOut(*Result,*Out);
   writeLine("stdout","Command result is");
   writeLine("stdout","*Out");
}
INPUT *Cmd="hello", *Arg="written"
OUTPUT ruleExecOut
```

"hello" is an executable script in iRODS/server/bin/cmd.

Example Policy Implementation

Using "asPostProcForPut" to implement policy, depending on resource

Data coming in to a target iRODS **resource** triggers a script that takes some desired action, triggers message to admin (unix) user

```
acPostProcForPut{ on($rescName like "demoResc") {
    writeLine("serverLog","USER, OBJPATH, and FILEPATH:
                $userNameClient, $objPath and $filePath");
    msiExecCmd("resource-trigger.sh","$rescName $objPath
                $userNameClient","null","null","null",*Out);
    msiSendMail("leesa@renci.org","resource $rescName","User
        $userNameClient just ingested file $objPath into
                $rescName.");
    }
}
```

- **acPostProcForPut** is contained in iRODS/server/config/reConfigs/core.re
- **resource-trigger.sh** is contained in /server/bin/cmd

Example script resource-trigger.sh

- > more resource-trigger.sh

```
#!/bin/sh
# echo "execCmdRule: "$execCmdRule

rescName=$1
objPath=$2
userNameClient=$3

echo "User $userNameClient just ingested file $objPath into
    $rescName"
echo "User $userNameClient just ingested file $objPath into
    $rescName" > /tmp/resource.out
```

Example Policy Implementation

Using asPostProcForPut to implement policy: inputs to a specific resource

Data coming in to a target iRODS **collection** triggers a script that takes some desired action (sending data to a remote ftp site)

```
acPostProcForPut{ on($objPath like "/compZone/home/outgoing/*") {
        writeLine("serverLog",
                          "$userNameClient sending $objPath to NCDC");
        msiSplitPath($filePath,*fileDir,*fileName);
        msiExecCmd("test_out.sh","*fileDir *fileName","null","null","null",*Out);
        msiSendMail("leesa@renci.org",
                  "send to NCDC","User $userNameClient
                                                  sent $objPath to NCDC.");
    }
}
```

- **acPostProcForPut** is contained in iRODS/server/config/reConfigs/core.re
- **acPostProcForPut** is the **same rule** in both examples! Just using different conditions.

Example script test_out.sh

- > more test_out.sh

```
#!/bin/sh
HOST=ftp.****.****.***
USER=anonymous
PASSWD=leesa@renci.org
srcDir=$1
srcFile=$2
echo $srcDir
echo $srcFile
#echo "Place holder for outgoing script. Dir: $srcDir, File: $srcFile"
echo "Place holder for outgoing script. Dir: $srcDir, File: $srcFile"
    > /tmp/test.out
```

Rules and Parameters

- Literals
  - constants: strings or numbers
  - a variable name not beginning with a special character (#, $ or *) is taken as string input
  - can only be used as input parameters (not output)
- Workflow variables
- Session state variables
- Persistent state variables

Workflow Variables (*variables)

- For example, in the following workflow chain:

```
myRule{
        msiDataObjOpen(*file,*FD);
        msiDataObjRead(*FD,10000,*BUF);
        writeLine("stdout",*BUF);
…
}

INPUT *file="/newZone/home/leesa/hello"
OUTPUT ruleExecOut
```

('stdout' is a structure managed by iRODS.)

- *file is an input parameter
- *FD is output from msiDataObjOpen and input to msiDataObjRead.
- *file, *FD, and *BUF are workflow variables

Session Variables ($variables)

- contain temporary information maintained during a server session
- contain information about client-server connection, data objects, user information, resource information, etc.
- contain information that can be sent back to the client. Example: stdout, stderr
- persistent across rule executions in the same session, so can be used to pass information between rule executions
- pre-defined by iRODS, stored as a complex C-structure (the rei structure)

Session Variables ($variables)

- $variables map to specific locations in this structure - mapping contained in server/config/reConfigs/**core.dvm**

- Example:

  $objPath | |rei->doi->objPath

  $objPath | |rei->doinp->objPath

  $dataType | |rei->doi->dataType

  $userNameClient | |rei->uoic->userName

  $collName | |rei->coi->collName

  $collParentName | |rei->coi->collParentName

  (Mappings are not necessarily unique.)

- https://www.irods.org/index.php/Session_State_Variables

Persistent State Variables (#variables)

- See iRODS Primer
- iRODS/lib/core/include/rodsGenQuery.h defines the attributes available via the General Query interface.
- Names begin with 'COL_' (column) for easy identification in the source code.
- "iquest" uses these field names but without the COL_ prefix:
  iquest attrs
- https://www.irods.org/index.php/Persistent_State_Information_Variables

Rule Condition

- Boolean expression
- Examples
  1. Run if msiService succeeds:
     rule1 { on (msiService >= 0) { ... } }
  2. Run if resource is demoResc8:
     rule2{ { on ($rescName == demoResc8) {…} }
  3. Run if the pathname begins with /x/y/z:
     Rule3{ {on ($objPath like /x/y/z/*) {…} }
- Same rule can give different actions depending on which condition is met
- Many operators
  ==, !=, >, <, >=, <=
  %%, !! (and, or)
  expr like reg-expr , expr not like reg-expr , expr ::= string

Delayed Execution

- Example

```
myTestRule{
        delay("<PLUSET>1m</PLUSET>"){
        writeLine("stdout","Writing message with a delay.");
        msiSendStdoutAsEmail(*Mailto, "Sending email");
        }
}
INPUT *Mailto="leesa@renci.org"
OUTPUT ruleExecOut
```

- Queue management:
  - iqstat
  - iqdel
  - iqmod

Periodic Execution

Example

```
myTestRule {
# Input parameters are:
#    Source collection path
#    Target collection path
#    Optional target resource
#    Optional synchronization mode:  IRODS_TO_IRODS
# Output parameter is:
#    Status of the operation
# Output from running the example is:
#  Synchronized collection 1 with collection 2
#
  delay("<PLUSET>5m</PLUSET>EF>1h</EF>"){
   msiCollRsync(*srcColl,*destColl,*Resource,"IRODS_TO_IRODS",*Status);
   writeLine("stdout","Synchronized collection *srcColl with collection *destColl");
   }
}
INPUT *srcColl="/compZone/home/leesa/tutorials", *destColl="/compZone/home/
      leesa/tutorials2", *Resource="demoResc"
OUTPUT ruleExecOut
```

Listing the Rule Base

showCore.r rule (text file)

```
showCoreRules {
#  Listing of the core.re file
#
# Input parameters:
#  none
msiAdmShowCoreRE();
}
INPUT null
OUTPUT ruleExecOut
```

An admin user can execute the rule to show the rule base:

–irule –vF showCore.r

Out-of-the-Box Services

Microservices for…

- Queries on metadata catalog
- Interaction with web services
- Invocation of external applications
- Workflow constructs (loops, conditionals, exit)
- Remote and delayed execution control

Microservices

print_hello_arg
msiVacuum
msiQuota
msiGoodFailure
msiSetResource
msiCheckPermission
msiCheckOwner
msiCreateUser
msiCreateCollByAdmin
msiSendMail
recover_print_hello
msiCommit
msiRollback
msiDeleteCollByAdmin
msiDeleteUser
msiAddUserToGroup
msiSetDefaultResc
msiSetRescSortScheme
msiSysReplDataObj
msiStageDataObj
msiSetDataObjPreferredResc
msiSetDataObjAvoidResc
msiSortDataObj
msiSysChksumDataObj
msiSetDataTypeFromExt
msiSetNoDirectRescInp
msiSetNumThreads
msiDeleteDisallowed
msiOprDisallowed

msiDataObjCreate
msiDataObjOpen
msiDataObjClose
msiDataObjLseek
msiDataObjRead
msiDataObjWrite
msiDataObjUnlink
msiDataObjRepl
msiDataObjCopy
msiExtractNaraMetadata
msiSetMultiReplPerResc
msiAdmChangeCoreIRB
msiAdmShowIRB
msiAdmShowDVM
msiAdmShowFNM
msiAdmAppendToTopOfCoreIRB
msiAdmClearAppRuleStruct
msiAdmAddAppRuleStruct
msiGetObjType
msiAssociateKeyValuePairsToObj
msiExtractTemplateMDFromBuf
msiReadMDTemplateIntoTagStruct
msiDataObjPut
msiDataObjGet
msiDataObjChksum
msiDataObjPhymv
msiDataObjRename
msiDataObjTrim
msiCollCreate

msiRmColl
msiReplColl
msiCollRepl
msiPhyPathReg
msiObjStat
msiDataObjRsync
msiFreeBuffer
msiNoChkFilePathPerm
msiNoTrashCan
msiSetPublicUserOpr
whileExec
forExec
delayExec
remoteExec
forEachExec
msiSleep
writeString
writeLine
writeBytesBuf
writePosInt
writeKeyValPairs
msiGetDiffTime
msiGetSystemTime
msiHumanToSystemTime
msiStrToBytesBuf
msiApplyDCMetadataTemplate
msiListEnabledMS
msiSendStdoutAsEmail
msiPrintKeyValPair

msiGetValByKey
msiAddKeyVal
assign
ifExec
break
applyAllRules
msiExecStrCondQuery
msiExecStrCondQueryWithOptions
msiExecGenQuery
msiMakeQuery
msiMakeGenQuery
msiGetMoreRows
msiAddSelectFieldToGenQuery
msiAddConditionToGenQuery
msiPrintGenQueryOutToBuffer
msiExecCmd
msiSetGraftPathScheme
msiSetRandomScheme
msiCheckHostAccessControl
msiGetIcatTime
msiGetTaggedValueFromString
msiXmsgServerConnect
msiXmsgCreateStream
msiCreateXmsgInp
msiSendXmsg
msiRcvXmsg
msiXmsgServerDisConnect
msiString2KeyValPair
msiStrArray2String
msiRdaToStdout

Microservices

msiRdaToDataObj
msiRdaNoResults
msiRdaCommit
msiAW1
msiRdaRollback
msiRenameLocalZone
msiRenameCollection
msiAclPolicy
msiRemoveKeyValuePairsFromObj
msiDataObjPutWithOptions
msiDataObjReplWithOptions
msiDataObjChksumWithOptions
msiDataObjGetWithOptions
msiSetReServerNumProc
msiGetStdoutInExecCmdOut
msiGetStderrInExecCmdOut
msiAddKeyValToMspStr
msiPrintGenQueryInp
msiTarFileExtract
msiTarFileCreate
msiPhyBundleColl
msiWriteRodsLog
msiServerMonPerf
msiFlushMonStat
msiDigestMonStat
msiSplitPath
msiGetSessionVarValue
msiAutoReplicateService

msiDataObjAutoMove
msiGetContInxFromGenQueryOut
msiSetACL
msiSetRescQuotaPolicy
msiPropertiesNew
msiPropertiesClear
msiPropertiesClone
msiPropertiesAdd
msiPropertiesRemove
msiPropertiesGet
msiPropertiesSet
msiPropertiesExists
msiPropertiesToString
msiPropertiesFromString
msiRecursiveCollCopy
msiGetDataObjACL
msiGetCollectionACL
msiGetDataObjAVUs
msiGetDataObjPSmeta
msiGetCollectionPSmeta
msiGetDataObjAIP
msiLoadMetadataFromDataObj
msiExportRecursiveCollMeta
msiCopyAVUMetadata
msiGetUserInfo
msiGetUserACL
msiCreateUserAccountsFromDataObj
msiLoadUserModsFromDataObj

msiDeleteUsersFromDataObj
msiLoadACLFromDataObj
msiGetAuditTrailInfoByUserID
msiGetAuditTrailInfoByObjectID
msiGetAuditTrailInfoByActionID
msiGetAuditTrailInfoByKeywords
msiGetAuditTrailInfoByTimeStamp
msiSetDataType
msiGuessDataType
msiMergeDataCopies
msiIsColl
msiIsData
msiGetCollectionContentsReport
msiGetCollectionSize
msiStructFileBundle
msiCollectionSpider
msiFlagDataObjwithAVU
msiFlagInfectedObjs

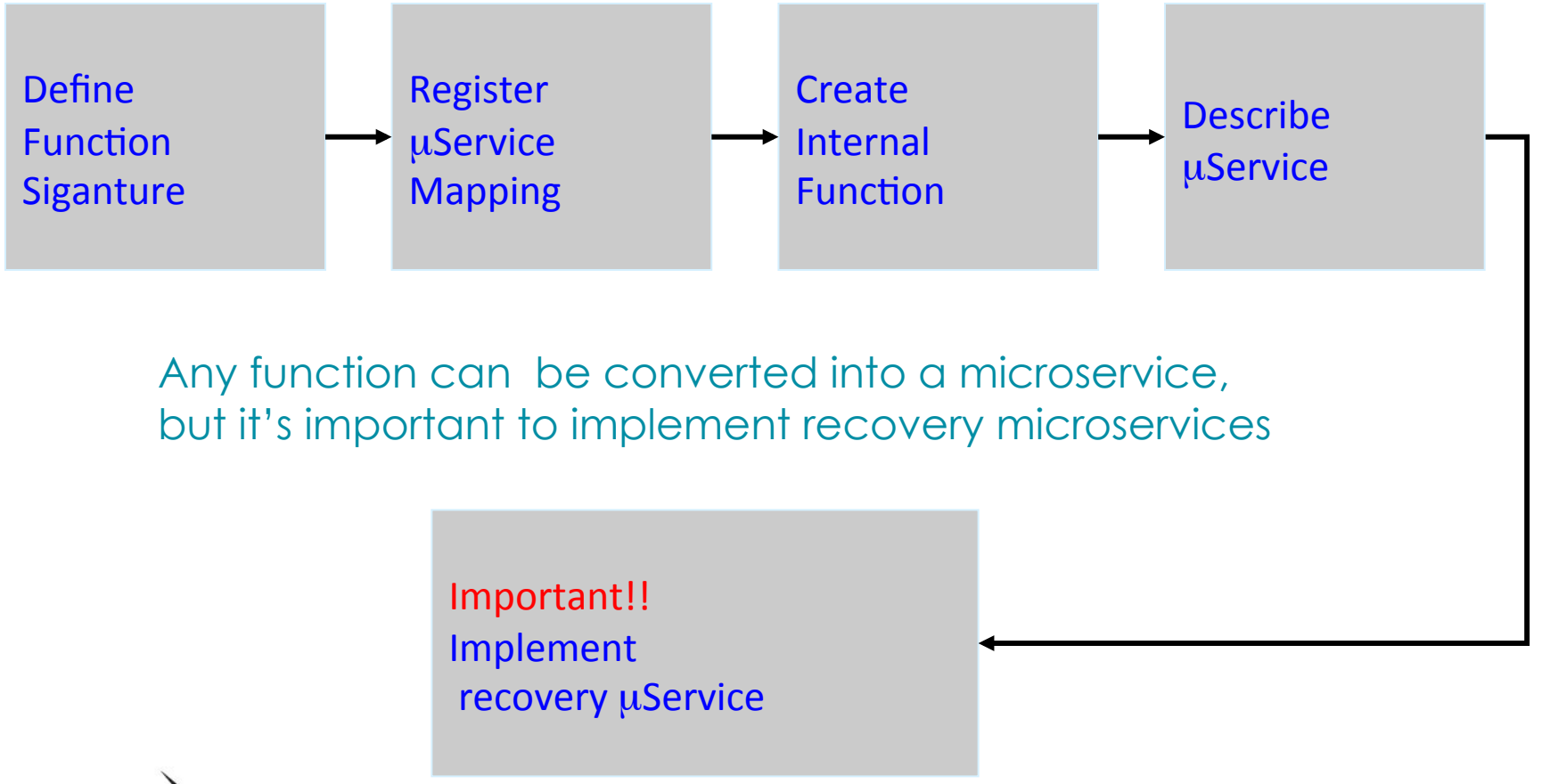
Microservice Modules

- Must be compiled with the code
- Consult the Microservice book: The integrated Rule-Oriented Data System (iRODS) Micro-service Workbook to see which module a microservice is contained in
- Enable that module: Enabled… yes in info.txt
- Example
  - > irule -F rulemsiCopyAVUMetadata.r

    ERROR: rcExecMyRule error. status = -1102000 NO_MICROSERVICE_FOUND_ERR
    Level 0: DEBUG: execMicroService3: no micro service found
    line 12, col 2
      msiFlagDataObjwithAVU(*Source,*Flag,*Status);
  - msiFlagDataObjwithAVU is contained in module ERA
  - enable module ERA
  - ./irodsctl istop
  - ./irodssetup

Creating New Microservices

Modules



Xmsg – the messaging server

On the server host:

1. Edit iRODS/server/config/server.config
   - uncomment this line and choose a host for running the xmsg server:
     xmsgHost norstore-trd-irods0.hpc.ntnu.no
   - add this line just below the first one:
     xmsgPort 1237 (choose some available port)

2. Put these same two lines into .irodsEnv file of the account running the iRODS server, for example:
   - xmsgHost 'ischia.renci.org'
   - xmsgPort 1237

   ← (Quotation marks are correct here.)

3. Recompile and restart
   - ./irodsctl istop (!!!!!!)
   - gmake clean
   - gmake (do NOT use irodssetup)

Xmsg – the messaging server

On the client side:

- Put these same two lines into .irodsEnv file of the client user
  - xmsgHost 'ischia.renci.org'
  - xmsgPort 1237

Now the two (server and client) can communicate:

- Admin user: ixmsg s –M "message to send"
- Client user: ixmsg r
- One sends, the other receives

idbug – uses ixmsg to track events

See the policy points hit by an "event"

A tool for facilitating the implementation of policy

- On the server host:
  1. Edit iRODS/scripts/perl/irodsctl.pl: uncomment this line
     $GLOBALREDEBUGFLAG=4;
     norstore-trd-irods0.hpc.ntnu.no
  2. ./irodsctl irestart (from iRODS directory)

Then any user can set up two windows to track activity:

- In one window: idbug –C
- In the other window: any icommand or iRODS activity

idbug example

Tracking policy points hit with "ils"

idbug: PROCESS BEGIN at ischia.renci.org: 17629. Client connected from 129.241.21.138 at port 1257

idbug:ApplyRule: :acChkHostAccessControl

idbug: ExecAction: :acChkHostAccessControl()

idbug: GotRule: :acChkHostAccessControl

idbug: ExecRule: :acChkHostAccessControl

idbug: ExecRule: Done:acChkHostAccessControl

idbug: ExecAction: Done:acChkHostAccessControl()

idbug:ApplyRule: Done:acChkHostAccessControl

idbug:ApplyRule: :acSetPublicUserPolicy

idbug: ExecAction: :acSetPublicUserPolicy()

idbug: GotRule: :acSetPublicUserPolicy

idbug: ExecRule: :acSetPublicUserPolicy

(ACL policy Steps)

idbug: ExecRule: Done:acSetPublicUserPolicy

idbug: ExecAction: Done:acSetPublicUserPolicy()

idbug:ApplyRule: Done:acSetPublicUserPolicy

idbug:ApplyRule: :acAclPolicy

idbug: ExecAction: :acAclPolicy()

idbug: GotRule: :acAclPolicy

idbug: ExecRule: :acAclPolicy

idbug: ExecRule: Done:acAclPolicy

idbug: ExecAction: Done:acAclPolicy()

idbug:ApplyRule: Done:acAclPolicy

idbug: PROCESS END FROM ischia.renci.org:17629

idbug example

- Tracking policy points hit with "iput hello"
  - idbug: PROCESS BEGIN at ischia.renci.org:16903. Client connected from 152.54.1.123 at port 1250
  - idbug:ApplyRule: :acChkHostAccessControl
  - idbug: ExecAction: :acChkHostAccessControl()
  - idbug: GotRule: :acChkHostAccessControl
  - idbug: ExecRule: :acChkHostAccessControl
  - idbug: ExecRule: Done:acChkHostAccessControl
  - idbug: ExecAction: Done:acChkHostAccessControl()
  - idbug:ApplyRule: Done:acChkHostAccessControl
  - idbug:ApplyRule: :acSetPublicUserPolicy
  - idbug: ExecAction: :acSetPublicUserPolicy()
  - idbug: GotRule: :acSetPublicUserPolicy
  - idbug: ExecRule: :acSetPublicUserPolicy
  - idbug: ExecRule: Done:acSetPublicUserPolicy
  - idbug: ExecAction: Done:acSetPublicUserPolicy()
  - idbug:ApplyRule: Done:acSetPublicUserPolicy
  - idbug:ApplyRule: :acAclPolicy
  - idbug: ExecAction: :acAclPolicy()
  - idbug: GotRule: :acAclPolicy
  - idbug: ExecRule: :acAclPolicy
  - idbug: ExecAction: :msiAclPolicy(STRICT)
  - idbug: ExecMicroSrvc: :msiAclPolicy(STRICT)
  - idbug: ExecAction: Done:msiAclPolicy(STRICT)
  - idbug: ExecRule: Done:acAclPolicy
  - idbug: ExecAction: Done:acAclPolicy()
  - idbug:ApplyRule: Done:acAclPolicy

(ACL policy steps)

idbug example

- idbug:ApplyRule: :**acSetRescSchemeForCreate**
- idbug: ExecAction: :acSetRescSchemeForCreate()
- idbug: GotRule: :acSetRescSchemeForCreate
- idbug: ExecRule: :acSetRescSchemeForCreate
- idbug: ExecAction: :msiSetDefaultResc(demoResc, null)
- idbug: ExecMicroSrvc: :msiSetDefaultResc(demoResc, null)
- idbug: ExecAction: Done:msiSetDefaultResc(demoResc, null)
- idbug: ExecRule: Done:acSetRescSchemeForCreate
- idbug: ExecAction: Done:acSetRescSchemeForCreate()
- idbug:ApplyRule: **Done:acSetRescSchemeForCreate**
- idbug:ApplyRule: :**acRescQuotaPolicy**
- idbug: ExecAction: :acRescQuotaPolicy()
- idbug: GotRule: :acRescQuotaPolicy
- idbug: ExecRule: :acRescQuotaPolicy
- idbug: ExecAction: :msiSetRescQuotaPolicy(off)
- idbug: ExecMicroSrvc: :msiSetRescQuotaPolicy(off)
- idbug: ExecAction: Done:msiSetRescQuotaPolicy(off)
- idbug: ExecRule: Done:acRescQuotaPolicy
- idbug: ExecAction: Done:acRescQuotaPolicy()
- idbug:ApplyRule: **Done:acRescQuotaPolicy**
- idbug:ApplyRule: :**acSetVaultPathPolicy**
- idbug: ExecAction: :acSetVaultPathPolicy()
- idbug: GotRule: :acSetVaultPathPolicy

idbug example

- idbug: ExecRule: :acSetVaultPathPolicy
- idbug: ExecAction: :msiSetGraftPathScheme(no, 1)
- idbug: ExecMicroSrvc: :msiSetGraftPathScheme(no, 1)
- idbug: ExecAction: Done:msiSetGraftPathScheme(no, 1)
- idbug: ExecRule: Done:acSetVaultPathPolicy
- idbug: ExecAction: Done:acSetVaultPathPolicy()
- idbug:ApplyRule: **Done:acSetVaultPathPolicy**
- idbug:ApplyRule: :**acPreProcForModifyDataObjMeta**
- idbug: ExecAction: :acPreProcForModifyDataObjMeta()
- idbug: GotRule: :acPreProcForModifyDataObjMeta
- idbug: ExecRule: :acPreProcForModifyDataObjMeta
- idbug: ExecRule: Done:acPreProcForModifyDataObjMeta
- idbug: ExecAction: Done:acPreProcForModifyDataObjMeta()
- idbug:ApplyRule: **Done:acPreProcForModifyDataObjMeta**
- idbug:ApplyRule: :**acPostProcForModifyDataObjMeta**
- idbug: ExecAction: :acPostProcForModifyDataObjMeta()
- idbug: GotRule: :acPostProcForModifyDataObjMeta
- idbug: ExecRule: :acPostProcForModifyDataObjMeta
- idbug: ExecRule: Done:acPostProcForModifyDataObjMeta
- idbug: ExecAction: Done:acPostProcForModifyDataObjMeta()
- idbug:ApplyRule: **Done:acPostProcForModifyDataObjMeta**

idbug example

- idbug:ApplyRule: :**acPostProcForCreate**
- idbug: ExecAction: :acPostProcForCreate()
- idbug: GotRule: :acPostProcForCreate
- idbug: ExecRule: :acPostProcForCreate
- idbug: ExecRule: Done:acPostProcForCreate
- idbug: ExecAction: Done:acPostProcForCreate()
- idbug:ApplyRule: **Done:acPostProcForCreate**
- idbug:ApplyRule: :**acPostProcForPut**
- idbug: ExecAction: :acPostProcForPut()
- idbug: GotRule: :acPostProcForPut
- idbug: ExecRule: :acPostProcForPut
- idbug: ExecAction: :like(comp523Resc, demoResc)
- idbug: ExecAction: Done:like(comp523Resc, demoResc)
- idbug: ExecRule: Done:acPostProcForPut
- idbug: GotRule: :acPostProcForPut
- idbug: ExecRule: :acPostProcForPut
- idbug: ExecAction: :like(/myZone/home/leesa/hello, /compZone/home/outgoing/*)
- idbug: ExecAction: Done:like(/myZone/home/leesa/hello, /compZone/home/outgoing/*)
- idbug: ExecRule: Done:acPostProcForPut
- idbug: ExecAction: Done:acPostProcForPut()
- idbug:ApplyRule: **Done:acPostProcForPut**
- idbug: PROCESS END FROM ischia.renci.org:16903

Audit tracking

Auditing is not on by default in iRODS, so must turn it on

- Edit
/opt/iRODS/iRODS/server/icat/src/icatMidLevelRoutines.c

- Set
  int auditEnabled=2;

- Compile and restart

- Use iquest and microservices to query the audit table

- See iCAT schema notes for audit table info:
  https://www.irods.org/index.php/icat_schema_notes

Database Resources

- Database Resource (DBR): a database (or similar tabular information) that can be queried and updated via SQL statements (or other, for non-SQL)
- Database object (DBO): an interface to a set of tables, typically a query that returns results
- Database Objects typically contain SQL
- Query results are stored to an iRODS data object, a DBO Results file (DBOR).
- idbo command – to access the external DB resource
- access controls imposed by iRODS ACLs
- https://www.irods.org/index.php/Database_Resources and https://www.irods.org/index.php/Database_Resource_Administration

Persistent State Information

ZONE_ID
ZONE_NAME
ZONE_TYPE
ZONE_CONNECTION
ZONE_COMMENT
ZONE_CREATE_TIME
ZONE_MODIFY_TIME
USER_ID
USER_NAME
USER_TYPE
USER_ZONE
USER_DN
USER_INFO
USER_COMMENT
USER_CREATE_TIME
USER_MODIFY_TIME
RESC_ID
RESC_NAME
RESC_ZONE_NAME
RESC_TYPE_NAME
RESC_CLASS_NAME
RESC_LOC
RESC_VAULT_PATH
RESC_FREE_SPACE
RESC_FREE_SPACE_TIME

RESC_INFO
RESC_COMMENT
RESC_CREATE_TIME
RESC_MODIFY_TIME
RESC_STATUS
DATA_ID
DATA_COLL_ID
DATA_NAME
DATA_REPL_NUM
DATA_VERSION
DATA_TYPE_NAME
DATA_SIZE
DATA_RESC_GROUP_NAME
DATA_RESC_NAME
DATA_PATH
DATA_OWNER_NAME
DATA_OWNER_ZONE
DATA_REPL_STATUS
DATA_STATUS
DATA_CHECKSUM
DATA_EXPIRY
DATA_MAP_ID
DATA_COMMENTS
DATA_CREATE_TIME
DATA_MODIFY_TIME

DATA_ACCESS_TYPE
DATA_ACCESS_NAME
DATA_TOKEN_NAMESPACE
DATA_ACCESS_USER_ID
DATA_ACCESS_DATA_ID
COLL_ID
COLL_NAME
COLL_PARENT_NAME
COLL_OWNER_NAME
COLL_OWNER_ZONE
COLL_MAP_ID
COLL_INHERITANCE
COLL_COMMENTS
COLL_CREATE_TIME
COLL_MODIFY_TIME
COLL_ACCESS_TYPE
COLL_ACCESS_NAME
COLL_TOKEN_NAMESPACE
COLL_ACCESS_USER_ID
COLL_ACCESS_COLL_ID
META_DATA_ATTR_NAME
META_DATA_ATTR_VALUE
META_DATA_ATTR_UNITS
META_DATA_ATTR_ID
META_DATA_CREATE_TIME

Persistent State Information

META_DATA_MODIFY_TIME
META_COLL_ATTR_NAME
META_COLL_ATTR_VALUE
META_COLL_ATTR_UNITS
META_COLL_ATTR_ID
META_NAMESPACE_COLL
META_NAMESPACE_DATA
META_NAMESPACE_RESC
META_NAMESPACE_USER
META_RESC_ATTR_NAME
META_RESC_ATTR_VALUE
META_RESC_ATTR_UNITS
META_RESC_ATTR_ID
META_USER_ATTR_NAME
META_USER_ATTR_VALUE
META_USER_ATTR_UNITS
META_USER_ATTR_ID
RESC_GROUP_RESC_ID
RESC_GROUP_NAME
USER_GROUP_ID
USER_GROUP_NAME
RULE_EXEC_ID
RULE_EXEC_NAME
RULE_EXEC_REI_FILE_PATH
RULE_EXEC_USER_NAME
RULE_EXEC_ADDRESS
RULE_EXEC_TIME
RULE_EXEC_FREQUENCY
RULE_EXEC_PRIORITY
RULE_EXEC_ESTIMATED_EXE_TIME
RULE_EXEC_NOTIFICATION_ADDR
RULE_EXEC_LAST_EXE_TIME
RULE_EXEC_STATUS
TOKEN_NAMESPACE
TOKEN_ID
TOKEN_NAME
TOKEN_VALUE
TOKEN_VALUE2
TOKEN_VALUE3
TOKEN_COMMENT
AUDIT_OBJ_ID
AUDIT_USER_ID
AUDIT_ACTION_ID
AUDIT_COMMENT
AUDIT_CREATE_TIME
AUDIT_MODIFY_TIME
SL_HOST_NAME
SL_RESC_NAME
SL_CPU_USED
SL_MEM_USED
SL_SWAP_USED
SL_RUNQ_LOAD
SL_DISK_SPACE
SL_NET_INPUT
SL_NET_OUTPUT
SL_CREATE_TIME
SLD_RESC_NAME
SLD_LOAD_FACTOR
SLD_CREATE_TIME